REMARKS

Claims 1, 7, 9-11, 15, 17 and 19-21 have been amended to more particularly claim the invention. Claims 1-21 are pending. Reconsideration and allowance are respectfully requested in view of the following remarks.

The specification has been objected to with respect to the identification of a UK patent application on page 7. The Examiner states that the recited number (9920914.8) does not match the application number (9928340.0). The Examiner has misunderstood the two citations. The reference to UK application 9928340.0 refers to the priority filing for the present application. The reference in the specification to UK application 9920914.8 refers to a different application from the same assignee which discloses more information concerning special relocations. The reference on page 7 of the specification is correct. For the record, the cited UK application 9920914 is now issued in the United States as U.S. Patent No. 6,687,899.

The present invention is concerned with a "lister" which takes an object code sequence as an input and through a conversion process known as disassembling produces a source code listing. Figure 2 shows that an original source code module 1 is converted by an assembler/compiler 2 into an object code module 3. Various object code modules 3 are then linked by a linker 4 to form the executable program code 5. This linking operation involves the generation of relocation instructions which denote a link time modification to the original object code module sequences and their expressions/values to generate the executable program code. The lister 8 of the present invention draws from the object code modules 3 and other inputs such as from library object files 6 or the executable program code 5 to reproduce the original source

code and recover the expressions/values of the original object code module sequences which were lost in the executable program code due to link time modification of object code modules (through, for example, a patching operation). Generation of the original source code is especially useful for testing and/or debugging code (see Specification, page 7, third paragraph).

Claims 1-4, 6-17 and 19-21 were rejected under 35 U.S.C. 103(a) as being unpatentable over Cahill in view of Brooks or Cahill in view of Noda. Applicant respectfully traverses.

Applicant reiterates an objection to the proposed combination of Cahill with either Brooks (or Noda). It is clear from a review of Cahill that it does not teach or suggest generating source code listing from an object code sequence as recited by each of the independent claims. Rather, Cahill teaches the generation of program independent assembly code from the object code. Applicant first asserts that one skilled in the art would have no reason to consider Cahill at all since it is directed to generating assembly code which is well recognized by those skilled in the art to be completely different and distinct from source code. Even more specifically, one skilled in the art, with consideration of Brooks or Noda and a need to obtain source code from disassembling object code, would have no reason or suggestion to also consider Cahill since it is focused on obtaining assembly code and does not achieve disassembly to a human readable source code. Lastly, Applicant submits that the combination of either Brooks or Noda with Cahill is improper because such a combination would destroy the key feature of Cahill for producing assembly code for modification. If the teachings of Brooks or Noda were combined with Cahill, a different code would be produced and the features of the Cahill disassembly operation which are unique to the generated assembly code would be lost. In view of the

foregoing, Applicant again asserts that the proposed art combination using Cahill is improper and should be withdrawn.

Even if the proposed combination of references is maintained, Applicant asserts that these references fail to teach or suggest each claim limitation.

In the Office Action, the Examiner alleges that the claims do not provide specific description of what the derived additional information consists of (see, page 15, for example). Applicant has amended the claims as noted herein to more specifically claim what is the additional information in the context of the claimed relocation instruction. More specifically, the additional information is the expressions/values of the original object code sequence which were lost due to the link time modification of that original object code sequence to create the executable program. In other words, these expressions/values were link time modified or rearranged through, for example, a patching operation at linking of the plural object code modules which gave rise to the relocation instruction that is present in the executable program.

The cited prior art generally discloses processes for disassembling object code. The Examiner has correctly conceded that Cahill's disassembly techniques fails to explicitly teach or suggest reading relocation instructions. It is the Examiner's belief that the reading operation with respect to relocation instructions is inherently performed by Cahill's path tracing operation. There is no teaching or suggestion in Cahill, however, for any operation to identify a "relocation instruction" which relates to a link time modification or rearrangement of the original object code or for a disassembly operation to derive the expressions/values of the original object code sequence which were lost due to the link time modification of that original object code sequence

to create the executable program. Path tracing as disclosed by Cahill is irrelevant to identifying a relocation instruction and the expressions/values of the original object code as claimed.

As noted by Applicant it is possible with the prior art to disassemble an instruction in object code but nonetheless fail to be able to recover with the listing operation the expressions/values which were link time modified or rearranged through the patching operation which gave rise to the relocation instruction (see, Specification, pages 2-3). The present invention as claimed addresses this issue by not only identifying the relocation instruction but also recovering these expressions/values to thus provide an identification of what in the original object code sequences was link time modified or rearranged. There is no teaching or suggestion in Cahill for the ability to recover such link time lost expressions/values relating to identified relocation instructions, and in fact Cahill fails to even mention or identify the problem of recovering in a disassembly operation those original object code expressions/values.

With respect to Brooks and Noda, again Applicant asserts that no teaching exists for the combination of identifying a relocation instruction and further disassembling that instruction to obtain the expressions/values for what was link time modified or rearranged through linking of original object code sequences. The Examiner has failed to show that Brooks or Noda teaches anything more than what Applicant described as being in the prior art on pages 2-3 of the specification. More specifically, once compiled an object code sequence can be disassembled and then reassembled to recover the object code sequence. If the object code sequence is originally created by linking separate object code modules such that "relocation instructions" are necessary to modify and/or rearrange the original object code modules through patching, a

conventional disassembly would only be able to recover final (i.e., post linking) expressions or values but could not recover the original object code expressions/values. Neither Brooks nor Noda recognize this issue and the Examiner has failed to show that Brooks or Noda are capable of recovering that original object code expressions/values for what was link time modified or rearranged through the patching operation.

In view of the foregoing, Applicant respectfully submits that the claimed invention distinguishes over the cited combination of art reference. Allowance of all pending claims is respectfully requested.

Applicant further specifically recites in claims 7, 9 and 10 certain types of expressions/values which are obtained in the claimed invention from the relocation instruction.

In claim 7, Applicant claims that the expressions/values are an arithmetic expression which was defined in the original object code prior to linking but was not present in the object code obtained after linking. The Examiner has failed to show that the cited prior art teaches or suggests an operation to identify a relocation instruction which relates to a link time modification or rearrangement of the original object code and then disassemble that information in context with the associated program instruction to derive the original object code arithmetic expression. The Examiner asserts on page 5 of the final Office Action an implicit or inherent teaching of the claimed invention. However, the branch or jump table functionality noted by the Examiner has not been shown to be able to derive the original object code arithmetic expression which was present before linking. Again, the prior art methods appear to disassemble an instruction in object code but fail to be able to recover the original object code arithmetic expressions which

were link time lost due to modifications or rearrangements (for example, through a relocation patching operation).

Next, with respect to claim 9, Applicant claims that the expressions/values are an operand value which was defined in the original object code prior to linking but was not present in the object code obtained after linking. The Examiner has failed to show that the cited prior art teaches or suggests an operation to identify a relocation instruction which relates to a link time modification or rearrangement of the original object code and then disassemble that information in context with the associated program instruction to derive the original object code operand values. The Examiner asserts on page 5 of the final Office Action that assembler directives meet the operand values limitation. No reasoning or explanation accompanies this unsupported conclusion and the Examiner has failed to show how the assembler directive teaching has anything to do with recovery of original object code operand values. Again, the prior art methods disassemble an instruction in object code but fail to be able to recover the operand values with respect to original object code which were link time lost due to modifications or rearrangements.

Lastly, with respect to claim 10, Applicant claims that the expressions/values are event information which was defined in the original object code prior to linking but was not present in the object code obtained after linking. The Examiner has failed to show that the cited prior art teaches or suggests an operation to identify a relocation instruction which relates to a link time modification or rearrangement of the original object code and then disassemble that information in context with the associated program instruction to derive the original object code event

information. The Examiner asserts on page 5 of the final Office Action that a signal handler meets the event information limitation. No reasoning or explanation is provided as to how that signal handler teaching relates to a relocation instruction in general or to the recovery of original object code event information. Again, the prior art methods disassemble an instruction in object code but fail to be able to recover the event information with respect to original object code which were link time lost due to modifications or rearrangements.

Turning next to claims 13-14, 20 and 21, Applicants more specifically claim that the relocation instruction is read to determine a certain type of relocation instruction and further deriving from that certain type the original object code expressions/values. In the Examiner's analysis, locating tags and tag types as disclosed by Cahill have been equated to the claimed "type" of relocation instruction. Given Applicant's claimed more specific recitation of relocation instruction types, this analysis is clearly incorrect. There is no teaching or suggestion in either Cahill or Brooks for the examination of a relocation instruction so as to determine its type and derive the expressions/values from that type.

Claim 5 was rejected under 35 U.S.C. 103(a) as being unpatentable over Cahill in view of Brooks and Brandes. Also, Claim 5 was rejected under 35 U.S.C. 103(a) as being unpatentable over Cahill in view of Noda and Brandes. Applicant traverses and asserts that claim 5 is patentable for at least the reasons discussed above with respect to claim 1.

Claim 18 was rejected under 35 U.S.C. 103(a) as being unpatentable over Cahill in view of Brooks and Clarke. Also, Claim 18 was rejected under 35 U.S.C. 103(a) as being

unpatentable over Cahill in view of Noda and Clarke. Applicant traverses and asserts that claim 18 is patentable for at least the reasons discussed above with respect to claim 11.

In view of the above, it is believed that this application is in condition for allowance, and such a Notice is respectfully requested.

Respectfully submitted,

JENKENS & GILCHRIST, P.C.

Date: 8/31/04

Andre M. Szuwalski
Registration No. 35,701

1445 Ross Avenue, Suite 3200
Dallas, Texas 75202-2799
(Direct) 214/855-4795
(Fax) 214/855-4300